An Introduction to ScaLAPACK

Tony Drummond
Lawrence Berkeley National Laboratory
LADrummond@lbl.gov

ACTS Collection

Berkeley Lab

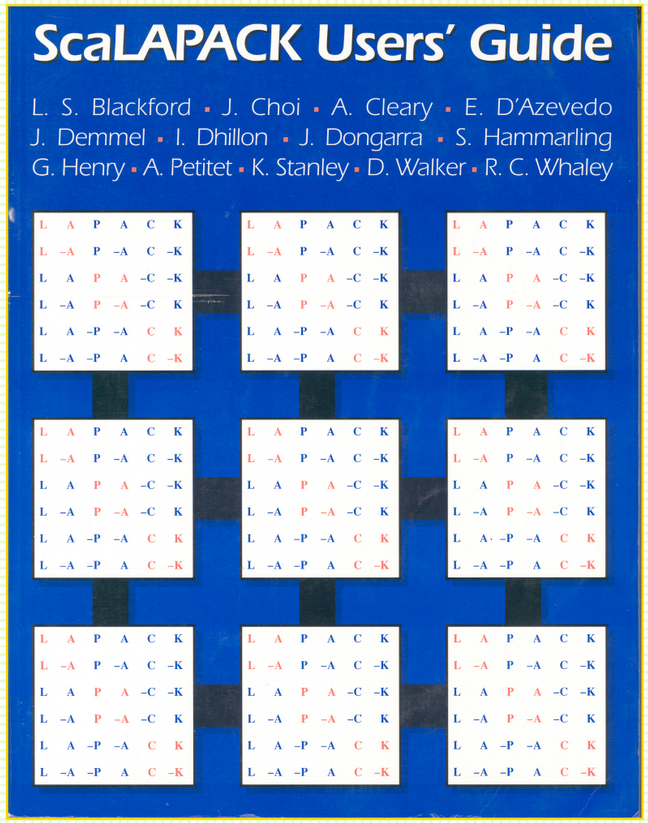
Office of Science
U.S. Department of Energy

An Introduction to ScaLAPACK

Outline:

- Introduction
- Applications
- ScaLAPACK Functionality
- Software Hierarchy and Interfaces
- ScaLAPACK User Interface
- Performance
- Afternoon: Hands-ON

An Introduction to ScaLAPACK



Team of Developers:

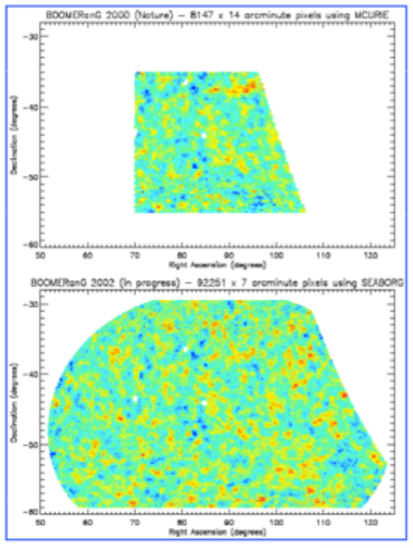
- Susan Blackford
- Jaeyoung Choi, Soongsil University
- Andy Cleary, LLNL
- Ed D'Azevedo, ORNL
- Jim Demmel, UCB
- Inderjit Dhillon, UT Austin
- Jack Dongarra, UTK
- Ray Fellers, LLNL
- Sven Hammarling, NAG
- Greg Henry, Intel
- Sherry Li, LBNL
- Osni Marques, LBNL
- Caroline Papadopoulos, UCSD
- Antoine Petitet, UTK
- Ken Stanley, UCB
- Francoise Tisseur, Manchester

MOTIVATION: Applications

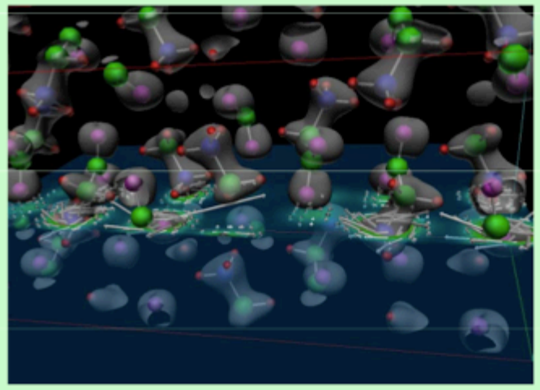
ACTS COLLECTION

- The statistics of the tiny variations in the CMB (the faint echo of the Big Bang) allows the determination of the fundamental parameters of cosmology to the percent level or better.
- MADCAP (Microwave Anisotropy Dataset Computational Analysis Package)
  - Makes maps from observations of the CMB and then calculates their angular power spectra. (See *http://crd.lbl.gov/~borrill*).
  - Calculations are dominated by the solution of linear systems of the form $M=A^{-1}B$ for dense nxn matrices A and B scaling as $O(n^3)$ in flops. MADCAP uses ScaLAPACK for those calculations.
- On a Cray T3E (original code):
  - Cholesky factorization and triangular solve.
  - Typically reached 70-80% peak performance.
  - Solution of systems with $n \sim 10^4$ using tens of processors.
  - The results demonstrated that the Universe is spatially flat, comprising 70% dark energy, 25% dark matter, and only 5% ordinary matter.
- On an IBM SP:
  - Porting was trivial but tests showed only 20-30% peak performance.
  - Code rewritten to use triangular matrix inversion and triangular matrix multiplication → one-day work !
  - Performance increased to 50-60% peak.
  - Solution of previously intractable systems with $n \sim 10^5$ using hundreds of processors.

The international BOOMERanG collaboration announced results of the most detailed measurement of the cosmic microwave background radiation (CMB), which strongly indicated that the universe is flat (Apr. 27, 2000).

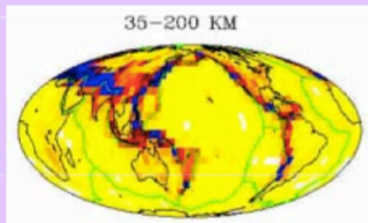


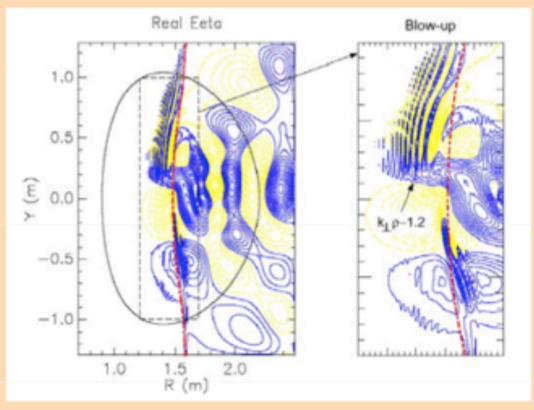
MOTIVATION: Applications



Induced current (white arrows) and charge density (colored plane and gray surface) in crystallized glycine due to an external field; courtesy of Louie, Yoon, Pfrommer and Canning (UCB and LBNL).

Model for the internal structure of the Earth, resolution matrix (Vasco and Marques)





Two ScaLAPACK routines, PZGETRF and PZGETRS, are used for solution of linear systems in the spectral algorithms based AORSA code (Batchelor et al.), which is intended for the study of electromagnetic wave-plasma interactions. The code reaches 68% of peak performance on 1936 processors of an IBM SP.

Advanced Computational Research in Fusion (SciDAC Project, PI Mitch Pindzola). Point of contact: Dario Mitnik (Dept. of Physics, Rollins College). Mitnik attended the workshop on the ACTS Collection in September 2000 and afterwards actively used ACTS tools, in particular ScaLAPACK. Dario has worked on the development, testing and support of new scientific simulation codes related to the study of atomic dynamics using time-dependent close coupling lattice and time-independent methods. He has reported that this work could not be carried out in sequential machines and that ScaLAPACK was fundamental for the parallelization of these codes.

An Introduction to ScaLAPACK

Outline:

- Introduction
- Applications
- ScaLAPACK Functionality
- Software Hierarchy and Interfaces
- ScaLAPACK User Interface
- Performance
- Numerical Reproducibility

ScaLAPACK's Functionality

$Ax = b$	Simple Driver	Expert Driver	Factor	Solve	Inversion	Conditioning Estimator	Iterative Refinement
Triangular				x	x	x	x
SPD	x	x	x	x	x	x	x
SPD Banded	x		x	x			
SPD Tridiagonal	x		x	x			
General	x	x	x	x	x	x	x
General Banded	x		x	x			
General Tridiagonal	x		x	x			
Least Squares	x		x	x			
GQR			x				
GRQ			x				
$Ax = \lambda x$ or $Ax = \lambda Bx$	Simple Driver	Expert Driver	Reduction	Solution			
Symmetric	x	x	x	x			
General	x	x	x	x			
Generalized BSPD	x		x	x			
SVD			x	x			

SPD: symmetric positive definite
GQR: generalized QR
GRQ: generalized RQ
BSPD: banded SPD
SVD: singular value decomposition

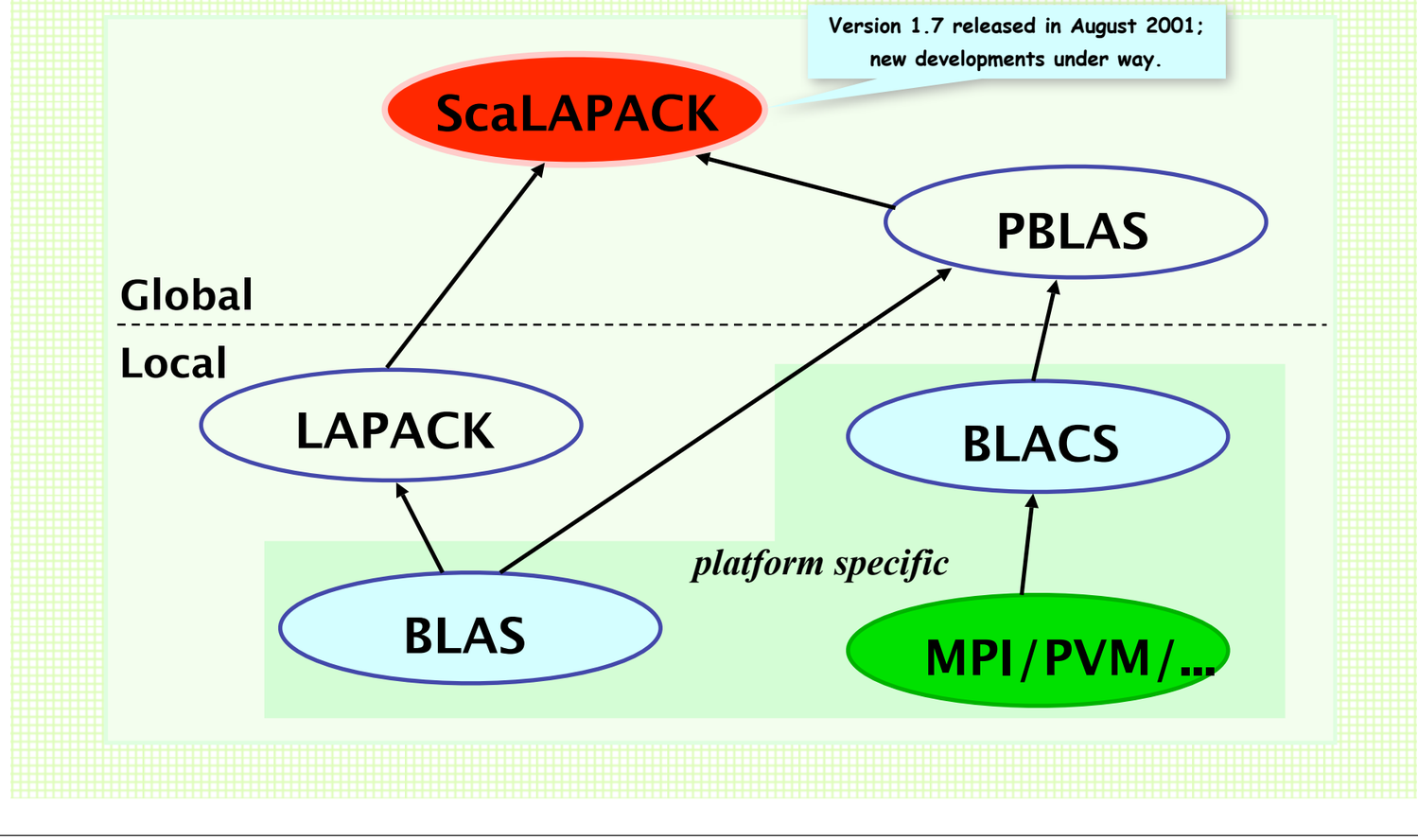
standard types of problems

distinct computational task

Outline:

- Introduction
- Applications
- ScaLAPACK Functionality
- Software Hierarchy and Interfaces
- ScaLAPACK User Interface
- Performance
- Afternoon: Hands-ON

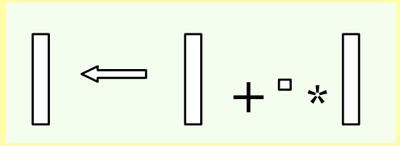
ScaLAPACK's Software Structure



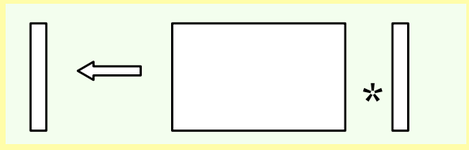
BLAS: Basic Linear Algebra Subroutines

BLAS LEVELS:

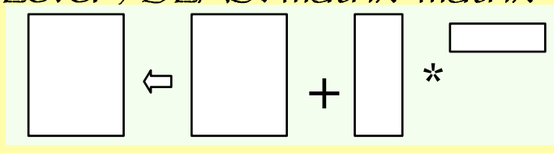
- Level 1 BLAS: vector-vector

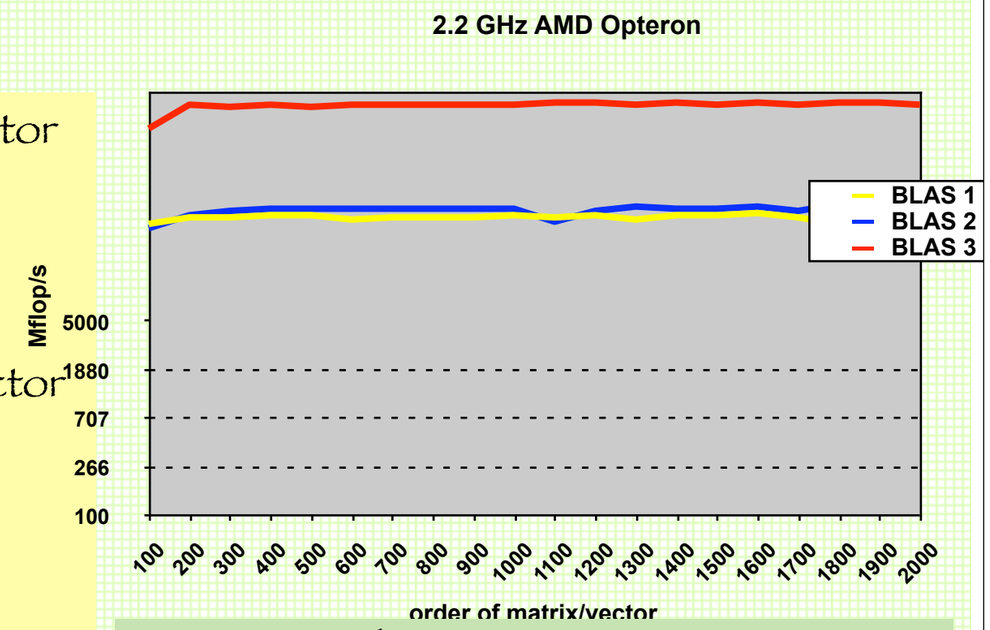


- Level 2 BLAS: matrix-vector



- Level 3 BLAS: matrix-matrix

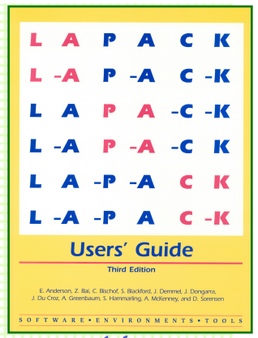




Design Considerations:

- Portability
- Performance: development of blocked algorithms is important for performance!

LAPACK: A Dense Linear Algebra Package



netlib.org

- Linear Algebra library written in Fortran 77 (Fortran 90)
- Combine algorithms from LINPACK and EISPACK into a single package.
- Efficient on a wide range of computers (RISC, Vector, SMPs).
- Built atop level 1, 2, and 3 BLAS Basic problems:
  - Linear systems: $Ax = b$
  - Least squares: $\min \|Ax - b\|_2$
  - Singular value decomposition: $A = U\Sigma V^T$
  - Eigenvalues and eigenvectors: $Az = \lambda z, \quad Az = \lambda Bz$
- LAPACK does not provide routines for structured problems or general sparse matrices (i.e. sparse storage formats such as compressed-row, -column, -diagonal, skyline ...).

BLACS:
Basic Linear Algebra Communication Subroutines

- Response to Message Passing based distributed communications
- Associate widely recognized mnemonic names with communication operations. This improves:
  - program readability
  - self-documenting quality of the code.
- Promote efficiency by identifying frequently occurring operations of linear algebra which can be optimized on various computers.

Basic Concepts of The BLACS Interface

- Promote efficiency by identifying common operations of linear algebra that can be optimized on various computers.
- Processes are embedded in a two-dimensional grid.

Example: a 3x4 grid

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11

- An operation which involves more than one sender and one receiver is called a scoped operation.

Scope	Meaning
Row	All processes in a process row participate.
Column	All processes in a process column participate.
All	All processes in the process grid participate.

BLACS Communication Routines

Send/Receive:

```
_xxSD2D(ICTXT,[UPLO,DIAG],M,N,A,LDA,RDEST,CDEST)
_xxRV2D(ICTXT,[UPLO,DIAG],M,N,A,LDA,RSRC,CSRC)
```

_ (Data type)	xx (Matrix type)
I: Integer, S: Real, D: Double Precision, C: Complex, Z: Double Complex.	GE: General rectangular matrix TR: Trapezoidal matrix

Broadcast:

```
_xxBS2D(ICTXT,SCOPE,TOP,[UPLO,DIAG],M,N,A,LDA)
_xxBR2D(ICTXT,SCOPE,TOP,[UPLO,DIAG],M,N,A,LDA,RSRC,CSRC)
```

SCOPE	TOP
‘Row’ ‘Column’ ‘All’	‘ ‘ (default) ‘Increasing Ring’ ‘1-tree’ …

BLACS Context

- BLACS context ⇔ MPI communicator
- The BLACS context is the BLACS mechanism for partitioning communication space.
- A message in a context cannot be sent or received in another context.
- The context allows the user to
  - create arbitrary groups of processes
  - create multiple overlapping and/or disjoint grids
  - isolate each process grid so that grids do not interfere with each other

An Example Code Using BLACS

```
*  Get system information
   CALL BLACS_PINFO( IAM, NPROCS )

*  Get default system context
   CALL BLACS_GET( 0, 0, ICTXT )

*  Define 1 x (NPROCS/2+1) process grid
   NPROW = 1
   NPCOL = NPROCS / 2 + 1
   CALL BLACS_GRIDINIT( ICTXT, 'Row', NPROW, NPCOL )
   CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )
*  If I'm not in the grid, go to end of program
   IF( MYROW.NE.-1 ) THEN
     IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
       CALL DGESD2D( ICTXT, 5, 1, X, 5, 1, 0 )
     ELSE IF( MYROW.EQ.1 .AND. MYCOL.EQ.0 ) THEN
       CALL DGERV2D( ICTXT, 5, 1, Y, 5, 0, 0 )
     END IF

     CALL BLACS_GRIDEXIT( ICTXT )
   END IF

   CALL BLACS_EXIT( 0 )
   END
```

(out) uniquely identifies each process
(out) number of processes available
(in) integer handle indicating the context
(in) use (default) system context
(out) BLACS context
(output) process row and column coordinate
send X to process (1,0)
receive X from process (0,0)
leave context
exit from the BLACS

PBLAS: Parallel BLAS

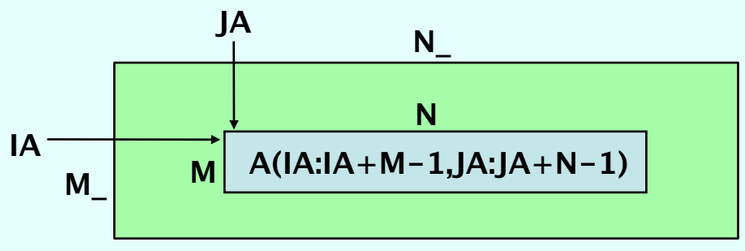
- Similar to the BLAS in portability, functionality and naming.
- Built atop the BLAS and BLACS
- Provide global view of matrix

```
CALL DGEXXX( M, N, A( IA, JA ), LDA, ... )
CALL PDGEXXX( M, N, A, IA, JA, DESCA, ... )
```

BLAS

PBLAS

Array descriptor (to be reviewed later)



An Introduction to ScaLAPACK

Outline:

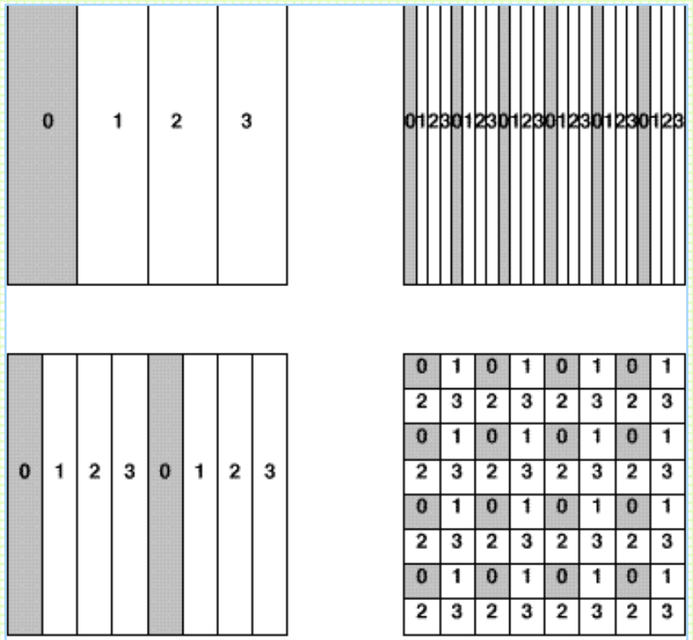
- Introduction
- Applications
- ScaLAPACK Functionality
- Software Hierarchy and Interfaces
- ScaLAPACK User Interface
- Performance Data
- Advanced Topics (OOC)

ScaLAPACK Design Goals

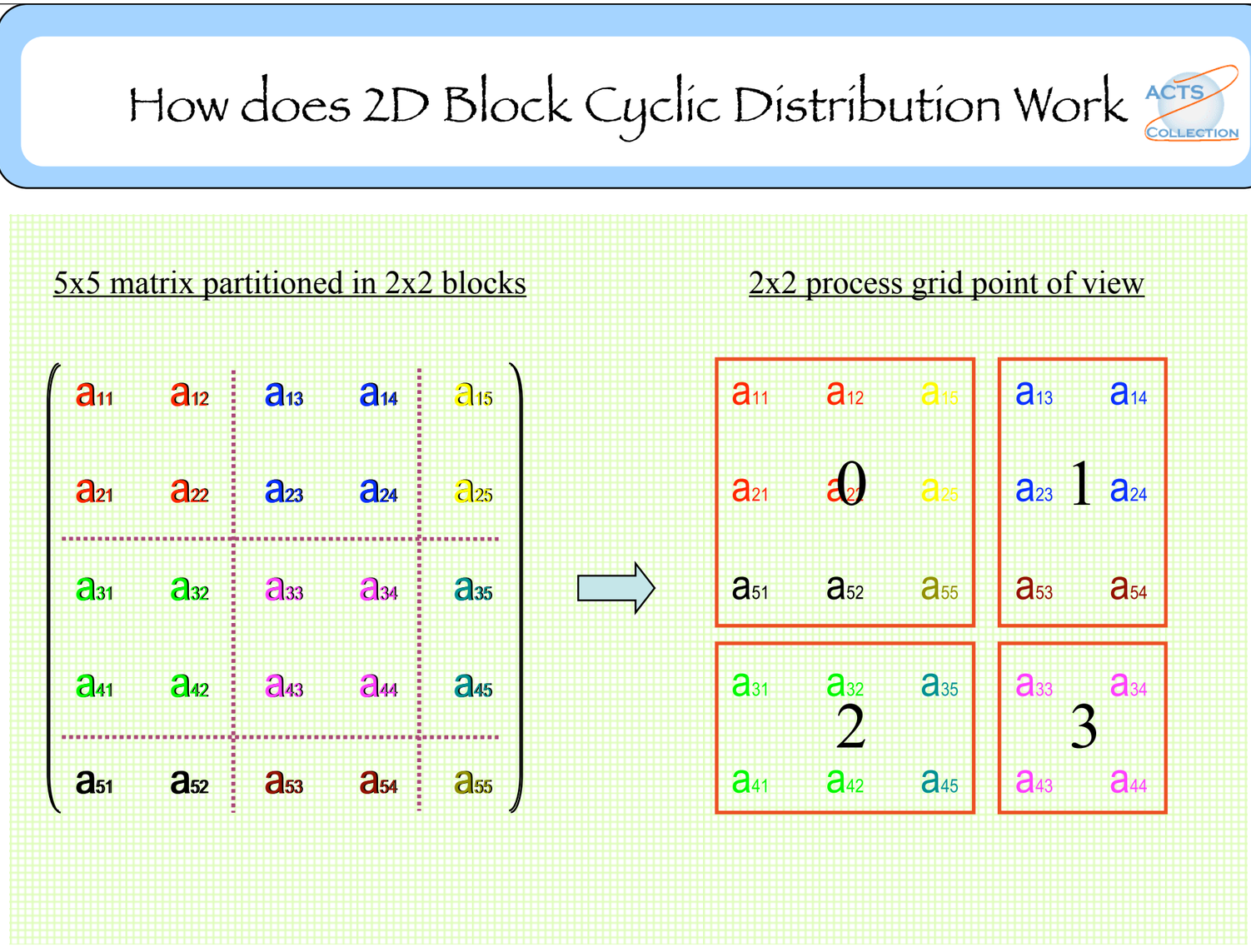
- Efficiency
  - Optimized computation and communication engines
  - Block-partitioned algorithms (Level 3 BLAS) for good node performance
- Reliability
  - Whenever possible, use LAPACK algorithms and error bounds.
- Scalability
  - As the problem size and number of processors grow
  - Replace LAPACK algorithm that did not scale (new ones into LAPACK)
- Portability
  - Isolate machine dependencies to BLAS and the BLACS
- Flexibility
  - Modularity: build rich set of linear algebra tools (BLAS, BLACS, PBLAS)
- Ease-of-Use
  - Calling interface similar to LAPACK

ScaLAPACK: Data Layouts

- 1D block and column distributions
- 1D block-cycle column and 2D block-cyclic distribution
- 2D block-cyclic distribution used in ScaLAPACK for dense matrices

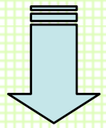


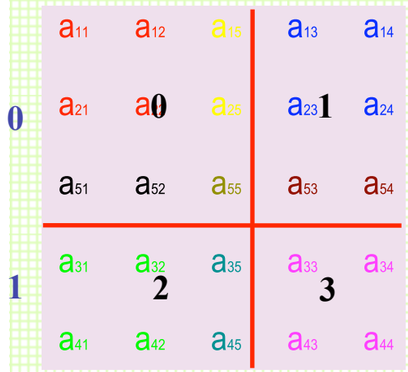
How does 2D Block Cyclic Distribution Work



An Example of 2D Block Cyclic Distribution

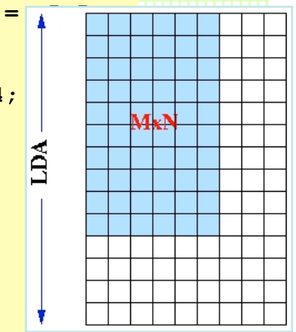
$$\begin{bmatrix} 1.1 & 1.2 & 1.3 & 1.4 & 1.5 \\ -2.1 & 2.2 & 2.3 & 2.4 & 2.5 \\ -3.1 & -3.2 & 3.3 & 3.4 & 3.5 \\ -4.1 & -4.2 & -4.3 & 4.4 & 4.5 \\ -5.1 & -5.2 & -5.3 & -5.4 & 5.5 \end{bmatrix}$$






```
 M


CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )


IF      ( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
        A(1) = 1.1; A(2) = -2.1; A(3) = -5.1;
        A(1+LDA) = 1.2; A(2+LDA) = 2.2; A(3+LDA) = -5.2;
        A(1+2*LDA) = 1.5; A(2+3*LDA) = 2.5; A(3+4*LDA) =
ELSE IF ( MYROW.EQ.0 .AND. MYCOL.EQ.1 ) THEN
        A(1) = 1.3; A(2) = 2.3; A(3) = -5.3;
        A(1+LDA) = 1.4; A(2+LDA) = 2.4; A(3+LDA) = -5.4;
ELSE IF ( MYROW.EQ.1 .AND. MYCOL.EQ.0 ) THEN
        A(1) = -3.1; A(2) = -4.1;
        A(1+LDA) = -3.2; A(2+LDA) = -4.2;
        A(1+2*LDA) = 3.5; A(2+3*LDA) = 4.5;
ELSE IF ( MYROW.EQ.1 .AND. MYCOL.EQ.1 ) THEN
        A(1) = 3.3; A(2) = -4.3;
        A(1+LDA) = 3.4; A(2+LDA) = 4.4;
END IF

  M


CALL PDGESVD( JOBU, JOBVT, M, N, A, IA, JA, DESCA, S, U, IU,
              JU, DESCU, VT, IVT, JVT, DESCVT, WORK, LWORK,
              INFO )
  M
```

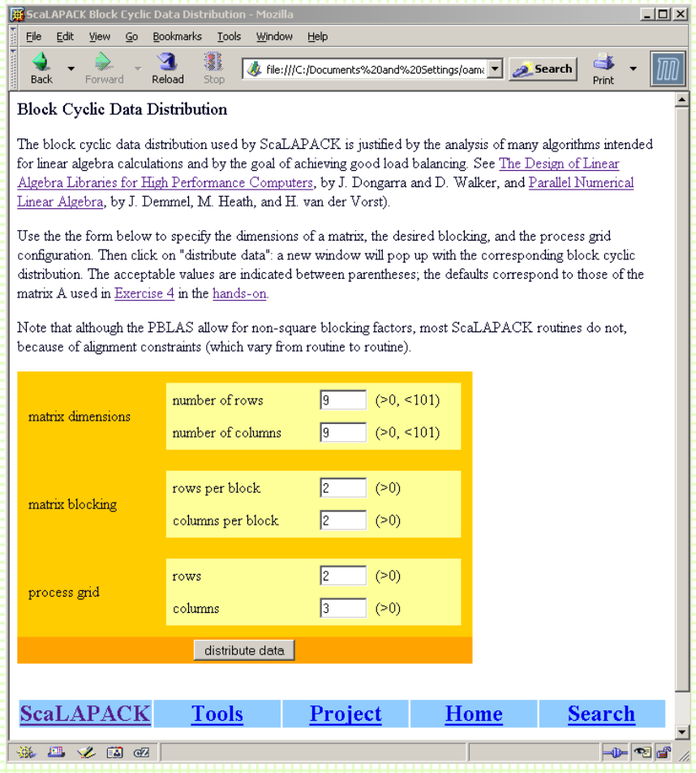
LDA is the leading dimension of the local array

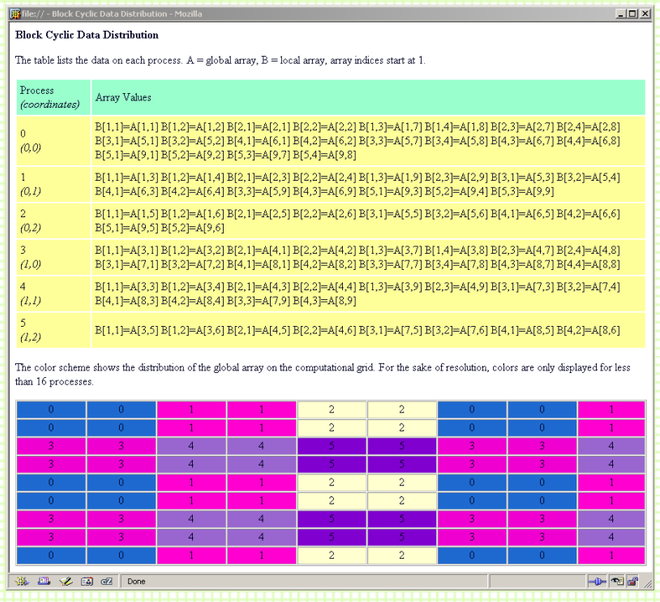


Why the headache of 2D block Cyclic Distribution?

- Ensures good load balance → performance and scalability (analysis of many algorithms to justify this layout).
- Encompasses a large number of data distribution schemes (but not all).
- Needs redistribution routines to go from one distribution to the other.
- See http://acts.nersc.gov/scalapack/hands-on/datadist.html

AID: http://acts.nersc.gov/scalapack/hands-on/datadist.html





ScaLAPACK: Array Descriptors

- Each global data object is assigned an array descriptor.
- The array descriptor:
  - Contains information required to establish mapping between a global array entry and its corresponding process and memory location (uses concept of BLACS context).
  - Is differentiated by the DTYPE_ (first entry) in the descriptor.
  - Provides a flexible framework to easily specify additional data distributions or matrix types.
- User must distribute all global arrays prior to the invocation of a ScaLAPACK routine, for example:
  - Each process generates its own submatrix.
  - One processor reads the matrix from a file and send pieces to other processors (may require message-passing for this).

Array Descriptor for Dense Matrices

DESC_()	Symbolic Name	Scope	Definition
1	DTYPE_A	(global)	Descriptor type DTYPE_A=1 for dense matrices.
2	CTXT_A	(global)	BLACS context handle.
3	M_A	(global)	Number of rows in global array A.
4	N_A	(global)	Number of columns in global array A.
5	MB_A	(global)	Blocking factor used to distribute the rows of array A.
6	NB_A	(global)	Blocking factor used to distribute the columns of array A.
7	RSRC_A	(global)	Process row over which the first row of the array A is distributed.
8	CSRC_A	(global)	Process column over which the first column of the array A is distributed.
9	LLD_A	(local)	Leading dimension of the local array.

Array Descriptor for Narrow Band Matrices

DESC_( )	Symbolic Name	Scope	Definition
1	DTYPE_A	(global)	Descriptor type DTYPE_A=501 for 1 x Pc process grid for band and tridiagonal matrices block-column distributed.
2	CTXT_A	(global)	BLACS context handle.
3	N_A	(global)	Number of columns in global array A.
4	NB_A	(global)	Blocking factor used to distribute the columns of array A.
5	CSRC_A	(global)	Process column over which the first column of the array A is distributed.
6	LLD_A	(local)	Leading dimension of the local array. For the tridiagonal subroutines, this entry is ignored.
7	–	–	Unused, reserved.

Array Descriptor for Right Hand Sides for Narrow Band Linear Solvers

DESC_( )	Symbolic Name	Scope	Definition
1	DTYPE_B	(global)	Descriptor type DTYPE_B=502 for Pr x 1 process grid for block-row distributed matrices
2	CTXT_B	(global)	BLACS context handle
3	M_B	(global)	Number of rows in global array B
4	MB_B	(global)	Blocking factor used to distribute the rows of array B
5	RSRC_B	(global)	Process row over which the first row of the array B is distributed
6	LLD_B	(local)	Leading dimension of the local array. For the tridiagonal subroutines, this entry is ignored
7	–	–	Unused, reserved

ScaLAPACK Routines

Three types of routines:

- Driver routines: each of which solves a complete problem, for example, solving a system of linear equations or computing the eigenvalues of a real symmetric matrix.
- Computational routines: each of which performs a distinct computational task, for example an LU factorization or the reduction of a real symmetric matrix to tridiagonal form. Each driver routine calls a sequence of computational routines. Global and local input error-checking are performed for these routines.
- Auxiliary routines: which in turn can be classified as follows:
  - routines that perform subtasks of block-partitioned algorithms -- in particular, routines that implement unblocked versions of the algorithms; and
  - routines that perform some commonly required low-level computations, for example, scaling a matrix, computing a matrix-norm, or generating an elementary Householder matrix.

ScaLAPACK Functionality

Ax = b	Driver type		Factor	Solve	Inversion	Conditioning estimator	Iterative Refinement
	Simple	Expert					
Triangular	x			x	x	x	x
SPD	x	x	x	x	x	x	x
SPD Banded	x		x	x			
SPD Tridiagonal	x		x	x			
General	x	x	x	x			
General Banded	x		x	x	x	x	x
General Tridiagonal	x		x	x			
Least Squares			x				
GQR	x		x	x			
GRQ			x				

Ax = λx or Ax = λBx	Simple	Expert	Reduce	Solve
Symmetric	x		x	x
General	x	x	x	x
Generalized BSPD	x	x	x	x
SVD			x	x

ScaLAPACK: Error Handling

- Driver and computational routines perform global and local input error-checking.
  - Global checking → synchronization
  - Local checking → validity
- No input error-checking is performed on the auxiliary routines.
- If an error is detected in a PBLAS or BLACS routine program execution stops.

ScaLAPACK: Debugging Hints

- Look at ScaLAPACK example programs.
- Always check the value of INFO on exit from a ScaLAPACK routine.
- Query for size of workspace, LWORK = –1.
- Link to the Debug Level 1 BLACS (specified by BLACSDBGLVL=1 in Bmake.inc).
- Consult errata files on netlib:

  http://www.netlib.org/scalapack/errata.scalapack

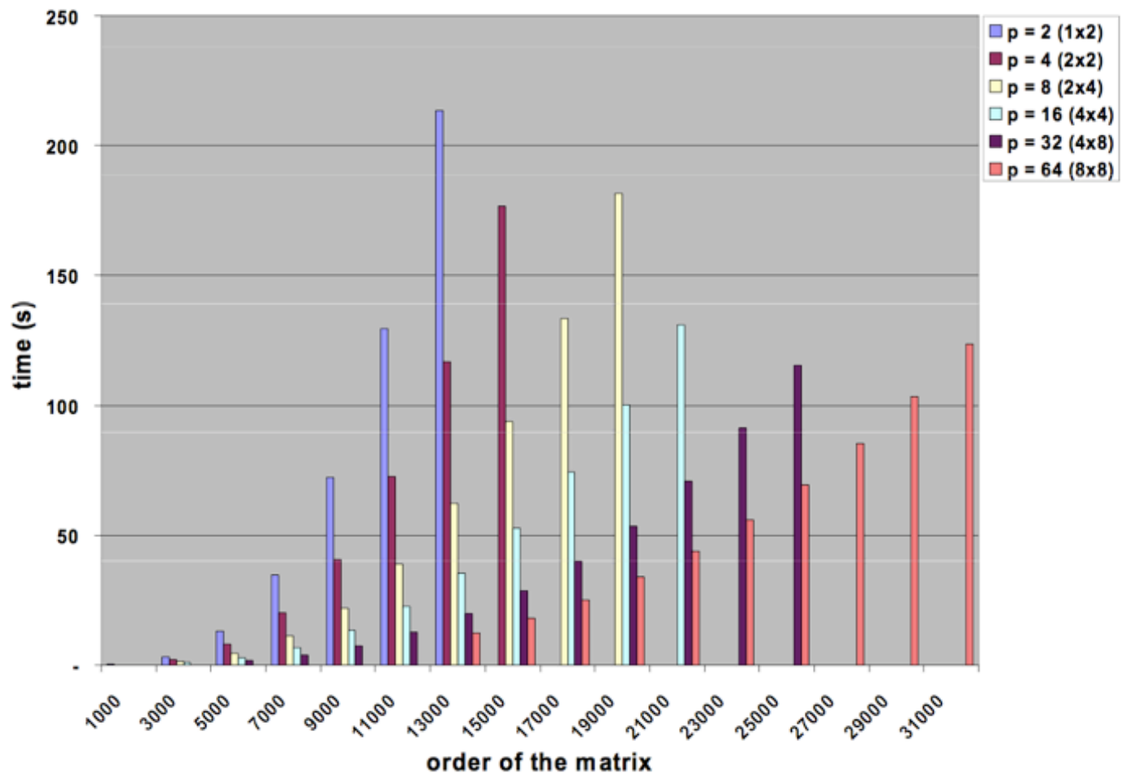
  http://www.netlib.org/blacs/errata.blacs

An Introduction to ScaLAPACK

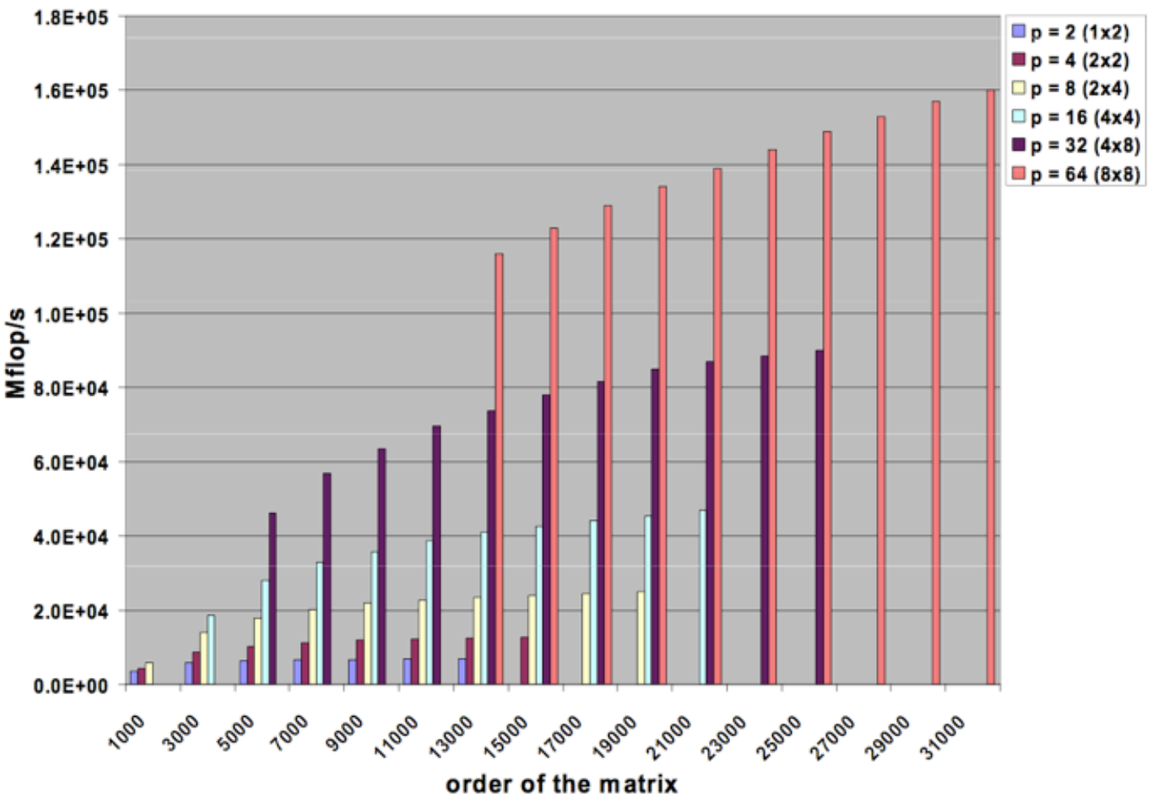
Outline:

- Introduction
- Applications
- ScaLAPACK Functionality
- Software Hierarchy and Interfaces
- ScaLAPACK User Interface
- Performance
- Advanced Topics (CR)

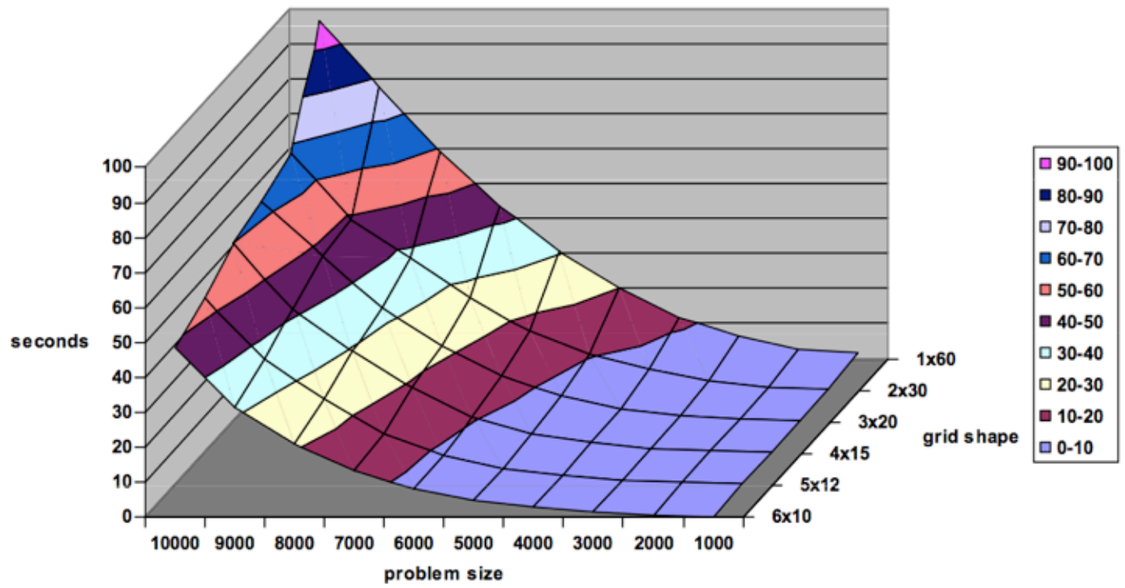
ScaLAPACK Performance



ScaLAPACK Performance



ScaLAPACK Performance



60 processors, Dual AMD Opteron 1.4GHz Cluster with Myrinet Interconnect, 2GB Memory

Commercial use of ScaLAPACK

ScaLAPACK has been incorporated in the following commercial packages:

- Fujitsu
- Hewlett-Packard
- Hitachi
- IBM Parallel ESSL
- NAG Numerical Library
- Cray LIBSCI
- NEC Scientific Software Library
- Sun Scientific Software Library
- Visual Numerics (IMSL)

Summary ScaLAPACK

- Library of high performance dense linear algebra routines for distributed-memory computing
- Reliable scalableandportable
- Reliable, scalable and portable
- Calling interface similar to LAPACK
- New developments on the way
- LAPACK/ScaLAPACK Forum: http://icl.cs.utk.edu/lapack-forum
- ScaLAPACK Users' Guide: http://www.netlib.org/scalapack/slug